LISTING OF CLAIMS

The listing of claims provided below replaces all prior versions and listing of claims in the application.

Please amend the claims as follows:

1. (currently amended): A method for loop optimization within a dynamic compiler system, comprising:

executing interpreted byte codes of a computer program having a loop structure, wherein the loop structure includes a loop exit test to be performed during each loop iteration and counting a number of times each of the interpreted byte codes is executed;

compiling the loop structure during the execution of the computer program ~~based on the loop structure being often used~~; and

creating an unrolled loop structure during the compiling operation, the creating of the unrolled loop structure being initiated by interpreted byte codes that are often used, as identified by the number of times each of the interpreted byte codes is executed, and wherein the unrolled loop structure includes a plurality of loop bodies based on the ~~original~~ loop structure.

2. (original): A method as recited in claim 1, wherein the unrolled loop structure includes the loop exit test.

3. (original): A method as recited in claim 2, wherein the loop exit test is performed once for each iteration of the plurality of loop bodies.

4. (original): A method as recited in claim 1, further comprising the operation of building a loop tree based on loops included in the computer program.

5. (original): A method as recited in claim 4, wherein nested loops are represented in the loop tree as child nodes.

6. (original): A method as recited in claim 5, wherein parallel loops are represented in the loop tree as nodes on a same level of the loop tree.

7. (original): A method as recited in claim 1, further including the operation of performing loop clean up.

8. (original): A method as recited in claim 7, wherein the loop clean-up includes optimizing multiple fall-in loop structures.

9. (original): A method as recited in claim 7, wherein the loop clean-up includes optimizing nested loop structures having invariant operations.

10. (currently amended): A dynamic compiling system, comprising:

an interpreter capable of interpreting ~~instructions~~ byte codes of a computer program during execution of the computer program, the interpreter being further capable of requesting that a particular ~~instruction~~ byte code be compiled ~~based on a predetermined number of repeated instructions~~, wherein the particular byte code is compiled based on a count of the number of times the particular byte code is executed; and

a compiler capable of compiling the ~~instructions~~ byte codes as requested by the interpreter, wherein the compiler is further capable of creating an unrolled loop structure when compiling an original loop structure of the computer program, wherein the unrolled loop structure includes a plurality of loop bodies based on the original loop structure.

11. (original): A dynamic compiling system as recited in claim 10, wherein the unrolled loop structure includes the loop exit test.

12. (original): A dynamic compiling system as recited in claim 11, wherein the loop exit test is performed once for each iteration of the plurality of loop bodies.

13. (original): A dynamic compiling system as recited in claim 10, wherein the compiler is further capable of building a loop tree based on loops included in the computer program.

14. (original): A dynamic compiling system as recited in claim 13, wherein nested loops are represented in the loop tree as child nodes.

15. (original): A dynamic compiling system as recited in claim 14, wherein parallel loops are represented in the loop tree as nodes on a same level of the loop tree.

16. (currently amended): A computer program embodied on a computer readable medium for loop optimization within a dynamic compiling, comprising:

a code segment that interprets byte codes of a computer program having a loop structure, wherein the loop structure includes a loop exit test to be performed during each loop iteration and counting a number of times each of the byte codes is executed;

a code segment that compiles the loop structure during the execution of the computer program ~~based on the loop structure being often used~~; and

a code segment that creates an unrolled loop structure during the compiling operation, the creating of the unrolled loop structure being initiated by the byte codes that are often used, as identified by the number of times each of the byte codes is executed, and wherein the unrolled loop structure includes a plurality of loop bodies based on the ~~original~~ loop structure.

17. (original): A computer program as recited in claim 16, wherein the unrolled loop structure includes the loop exit test, and wherein the loop exit test is performed once for each iteration of the plurality of loop bodies.

18. (original): A computer program as recited in claim 17, further comprising a code segment that performs loop clean up.

19. (original): A computer program as recited in claim 18, wherein the loop clean-up includes optimizing multiple fall-in loop structures.

20. (original): A computer program as recited in claim 19, wherein the loop clean-up includes optimizing nested loop structures having invariant operations.